Project 3

Questions

1. What is the purpose of having an individual handler function for each exception/interrupt? (i.e., if all exceptions/interrupts were delivered to the same handler, what feature that exists in the current implementation could not be provided?)
    a. We build different TrapFrames for each trap, and that allows us to take a specific action for each trap, such as how we are printing a different message for each trap.

2. Did you have to do anything to make the `user/softint` program behave correctly? The grade script expects it to produce a general protection fault (trap 13), but `softint` code says `int $14`. *Why* should this produce interrupt vector 13? What happens if the kernel actually allows `softint int $14` instruction to invoke the kernel's page fault handler (which is interrupt vector 14)?
    a. We had to use 0 for the `dpl` value in the SETGATE macro for the page fault – this prevents the user program from calling the page fault directly, which if allowed is a security concern. The 0 privilege level means that only kernel code can call the page fault interrupt.

3. The break point test case will either generate a break point exception or a general protection fault depending on how you initialized the break point entry in the IDT. Why? How do you need to set it up in order to get the breakpoint exception to work as specified above and what incorrect setup would cause it to trigger a general protection fault?
    a. We had to use 3 for the `dpl` value in the SETGATE macro – this allows the user to call it, and made it work correctly. When we did it incorrectly with `dpl=0`, produced the general protection.
4. What do you think is the point of these mechanisms, particularly in light of what the `user/softint` test program does?
    a. The point is to provide security and control how the user applications can access the traps provided by the processor. These mechanisms prevent potentially malicious actions such as what the `softint` program is doing.